Apprentissage par renforcement

IFT-17587
Concepts avancés pour systèmes intelligents
Luc Lamontagne

Plan de la présentation

- Tâche d'apprentissage par renforcement
- Apprentissage passif – *TD-Learning* + *ADP*
- Apprentissage actif - *Q-Learning*

Apprentissage par renforcement

- L'agent apprend par ses expériences dans l'environnement.
- Pour chaque action → une récompense ou une pénalité.
  - Ce n'est pas de l'apprentissage supervisé car on ne lui donne pas la bonne réponse.
- But de l'agent
  - Apprendre comment agir sur plusieurs étapes pour cumuler la plus grande récompense.
- Nous allons voir deux approches
  - *Apprentissage passif : TD-Learning + ADP*
  - *Apprentissage actif : Q-Learning*.

Apprentissage par renforcement :

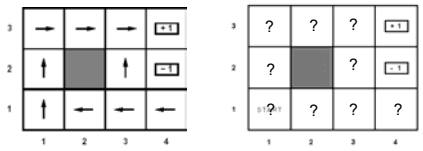
Tâche d'apprentissage

- L'agent peut percevoir l'environnement →*S* états.
- Il a un ensemble d'actions *A* qu'il peut exécuter.
- À chaque étape *t :*
  1. L'agent perçoit l'état courant s_t
  2. Il choisit l'action courante a_t et l'exécute.
  3. L'environnement lui accorde une récompense $r_t = r(s_t, a_t)$
  4. L'environnement produit le nouvel état $s_{t+1} = \delta(s_t, a_t)$
  5. L'agent répète les étapes 1-4 un certain nombre de fois et tente d'apprendre comment mieux agir (politique).
- Les fonctions *r* et δ appartiennent à l'environnement
  - Modélisé comme un MDP? Oui mais…
  - Ces fonctions ne sont pas nécessairement connues de l'agent.
  - A vrai dire… habituellement l'agent n'a pas de connaissance à priori de l'environnement.

Apprentissage par renforcement :

Apprentissage passif

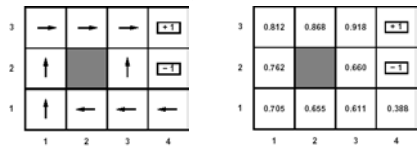
- Apprentissage passif = la politique π est fixe
  - Si état *s* → exécute l'action $\pi(s)$
- But : Déterminer la valeur de la politique π
  - Apprendre la fonction d'utilité $U^{\pi}(s)$
- Ça ressemble à une tâche d'évaluation de politique (MDP)
  - Par ex. l'algorithme *value iteration*
  - Mais on ne connait ni le modèle de transition *T(s,a,s')* ni la fonction *R(s)*



Apprentissage par renforcement :

Apprentissage passif

- L'agent exécute une série d'essais.
- Il perçoit la séquence d'états et les récompenses correspondantes.
  - Ces deux éléments sont gérés par l'environnement
  - Par exemple :
    $(1,1)_{-0.4} \rightarrow (1,2)_{-0.4} \rightarrow (1,3)_{-0.4} \rightarrow (1,2)_{-0.4} \rightarrow (1,3)_{-0.4} \rightarrow (2,3)_{-0.4} \rightarrow (3,3)_{-0.4} \rightarrow (4,3)_{+1}$
    $(1,1)_{-0.4} \rightarrow (1,2)_{-0.4} \rightarrow (1,3)_{-0.4} \rightarrow (2,3)_{-0.4} \rightarrow (3,3)_{-0.4} \rightarrow (3,2)_{-0.4} \rightarrow (3,3)_{-0.4} \rightarrow (4,3)_{+1}$
    $(1,1)_{-0.4} \rightarrow (2,1)_{-0.4} \rightarrow (3,1)_{-0.4} \rightarrow (3,2)_{-0.4} \rightarrow (4,2)_{-1}$
- Idée : Utiliser les récompenses obtenues des séquences d'entraînement pour apprendre les utilités espérées $U^{\pi}(s)$.
  - Utilité espérée = la somme moyenne des récompenses en suivant la politique



Apprentissage passif :

Programmation dynamique adaptative

- Idée :
  - On ne connait pas le modèle de transition *T(s, π(s), s')* …
  - Alors on tente de l'apprendre à partir des observations d'entraînement.
  - On cumule également les récompenses *R(s)*.
  - Et on utilise une méthode de programmation dynamique pour résoudre l'approximation de MDP
    - Par exemple, l'algorithme *policy iteration*.
- Performance de ADP :
  - Bonne pour les environnements totalement observables
  - Mais pratiquement impossible pour les grands espaces d'états
    - Ex : backgammon → résoudre 10^{50} équations !

Apprentissage passif :

Programmation dynamique adaptative

function PASSIVE-ADP-AGENT(*percept*) **returns** an action
  inputs: *percept*, a percept indicating the current state s' and reward signal r'
  static: π, a fixed policy
    mdp, an MDP with model $\mathcal{T}$, rewards R, discount γ
    U, a table of utilities, initially empty
    N_{sa}, a table of frequencies for state-action pairs, initially zero
    $N_{s'|sa}$, a table of frequencies for state-action-state triples, initially zero
    s, a, the previous state and action, initially null

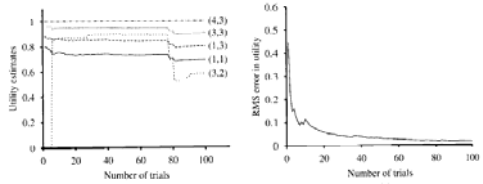
  if s' is new **then do** $U[s'] \leftarrow r'$; $R[s'] \leftarrow r'$
  if s is not null **then do**
    increment $N_{sa}[s, a]$ and $N_{s'|sa}[s', a, s]$
    for each t such that $N_{s'|sa}[t, a, s]$ is nonzero **do**
      $\mathcal{T}[s, a, t] \leftarrow N_{s'|sa}[s, a, t] / N_{sa}[s, a]$
  $U \leftarrow$ VALUE-DETERMINATION(π, U, *mdp*)
  if TERMINAL?(s') **then** $s, a \leftarrow$ **null else** $s, a \leftarrow s', \pi[s']$
  return a

Mise à jour *U* avec une fonction
Par ex. itération de politique

Apprentissage passif :

Programmation dynamique adaptative

- Exemple de convergence



Apprentissage passif :

TD-Learning

- Idée :
  - Utiliser les transitions observées s → s' durant l'entraînement pour ajuster les valeurs des états observés (une approximation).
  - Les valeurs devraient être cohérentes avec les équations de contraintes
  - Pas d'apprentissage de modèle de transitions.
- Mise à jour

$$U^{\pi}(s) \leftarrow U^{\pi}(s) + \alpha\,(R(s) + \gamma\, U^{\pi}(s') - U^{\pi}(s))$$

  - α : le paramètre du taux d'apprentissage
  - γ : le facteur d'escompte.
- *TD = temporal difference*
  - Utilise les différence d'utilités entre états successifs.
- Convergence
  - Si α décroit avec le temps, alors la convergence de *U(s)* est assurée.
  - Ne converge pas aussi rapidement que ADP mais beaucoup plus simple !

Apprentissage passif :

TD-Learning

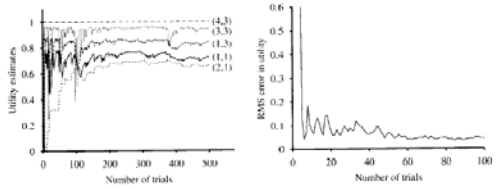
function PASSIVE-TD-AGENT(*percept*) **returns** an action
  inputs: *percept*, a percept indicating the current state s' and reward signal r'
  static: π, a fixed policy
    U, a table of utilities, initially empty
    N_s, a table of frequencies for states, initially zero
    s, a, r, the previous state, action, and reward, initially null

  if s' is new **then** $U[s'] \leftarrow r'$
  if s is not null **then do**
    increment $N_s[s]$
    $U[s] \leftarrow U[s] + \alpha(N_s[s])(r + \gamma U[s'] - U[s])$
  if TERMINAL?$[s']$ **then** $s, a, r \leftarrow$ null **else** $s, a, r \leftarrow s', \pi[s'], r'$
  return a

Apprentissage passif :

TD-Learning

- Exemple de convergence

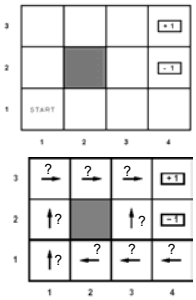


Apprentissage par renforcement :

Apprentissage actif

- Apprentissage actif
  - la politique π n'est pas fixe.
  - Apprendre une politique pour sélectionner la prochaine action a_t en se basant sur l'état courant s_t

    $\pi : S \rightarrow A$
- Méthode étudiée
  - *Q-Learning*
  - Il existe également une version active de ADP
    - pas étudiée dans cette prestation de cours.



Apprentissage actif :

Q-Learning

- Apprendre la politique optimale
  - c.-à-d. celle qui maximise la somme des récompenses pour tous les états *s*.

    $\pi^* \equiv \max_{\pi} V^{\pi}(s), (\forall s)$
- L'agent apprend une fonction Q
  - La qualité d'une action pour un état donné
- Elle représente

  la récompense immédiate obtenue en exécutant l'action *a* dans l'état *s*

  +

  la valeur obtenue en suivant la politique optimale par la suite.

  $Q(a,s) \leftarrow Q(a,s) + \alpha\,(R(s) + \gamma\, max_{a'} Q(a',s') - Q(a,s))$
- Lien entre utilités et Q-valeurs

  $U(s) = max_a\ Q(a,s)$

Apprentissage actif :

Q-Learning

function Q-LEARNING-AGENT(*percept*) **returns** an action
  inputs: *percept*, a percept indicating the current state s' and reward signal r'
  static: Q, a table of action values index by state and action
    N_{sa}, a table of frequencies for state-action pairs
    s, a, r, the previous state, action, and reward, initially null

  if s is not null **then do**
    increment $N_{sa}[s, a]$
    $Q[s, a] \leftarrow Q[s, a] + \alpha(N_{sa}[s, a])(r + \gamma \max_{a'} Q[s', a'] - Q[s, a])$
  if TERMINAL?[s'] **then** $s, a, r \leftarrow$ null
  else $s, a, r \leftarrow s', \text{argmax}_{a'} f(Q[s', a'], N_{sa}[s', a']), r'$
  return a

Apprentissage par renforcement :

Stratégie d'exploration

- Si l'agent choisit toujours l'action qui maximise Q
  - Il va avoir tendance à toujours prendre le même chemin.
  - Il n'explorera pas les autres possibilités qui sont peut-être meilleures.
- L'exploration permet d'étendre le modèle de l'environnement.
- Cependant il y a un compromis entre l'exploration et l'exploitation.
  - Exploration : découvrir de nouvelles options.
  - Exploitation : mettre en pratique les options connues.

Apprentissage par renforcement :

Stratégie d'exploration

- Pour favoriser l'exploration, on choisit aléatoirement une action
  - Mais en donnant plus de chance aux actions ayant une grande valeur de *Q*.

$$P(a_i|s) = \frac{k^{\bar{Q}(s,a_i)}}{\sum_j k^{\bar{Q}(s,a_j)}}$$

- $k > 0$ est une constante
  - Plus k est grand, plus on favorise les valeurs de *Q* élevées.
- En générale, on commence avec une petite valeur pour k et on l'augmente graduellement
  - Donc, il y a plus d'exploration au début.

Conclusion

- L'apprentissage par renforcement s'appuie sur le paradigme MDP.
- Il est utile lorsqu'on ne connaît pas le modèle de l'environnement.
- L'apprentissage passif de type *temporal difference* (*TD-Learning*) s'applique pour des politiques fixes.
  - Elle permet de faire converger les estimations de valeurs sans connaître le modèle de l'environnement.
  - D'autres méthodes, comme la programmation dynamique adaptative, permettent d'apprendre le modèle de transition.
- L'apprentissage actif de type *Q-Learning* guide le choix des actions pour un état donné.
  - On apprend donc la politique.
- Il est important d'amener l'algorithme à explorer pour découvrir de nouvelles pistes de solutions.
- Pour les espaces d'états très grands, il faut des techniques qui approximent les estimations d'utilités (pas étudié dans ce cours).
  - Réseaux de neurones.
  - Raisonnement à base d'exemples (kNN).